arXiv:1201.1979v2 [stat.ME] 8 May 2012

Clustering by self-updating process

Shang-Ying Shiu
Department of Statistics
National Taipei University, Taiwan
and
Ting-Li Chen
Institute of Statistical Science
Academia Sinica, Taiwan
email: tlchen@stat.sinica.edu.tw

May 9, 2012

Abstract

We introduce a simple, intuitive, yet powerful algorithm for clustering analysis. This algorithm stands from the viewpoint of elements to be clustered, and simulates the process of how they perform self-clustering. The algorithm is therefore named Self-Updating Process (SUP). We discover the algorithm's ability to simultaneously isolate noise while performing clustering, which enables the algorithm to produce good clustering results even when the level of noise in the data is high. We present simulation studies to demonstrate the performance of this algorithm. Applications to gene expression data and image segmentation are provided.

KEYWORDS: clustering, hierarchical clustering, k-means, mean-shift

1 Introduction

Clustering analysis is a useful technique to discover patterns in data. This technique has been widely applied to many disciplines for partitioning data into several groups. Within each group, elements are considered to resemble each other. In image segmentation the clustering technique is applied to partition an image into regions, each of which has its own color patterns. In biology and medicine, the technique can be used to classify subjects on the basis of their clinical responses. The resulting grouping structure provides valuable information to discover subtypes of a disease. With the breakthrough in experimental molecular biology, clustering has rapidly received a considerable amount of attention and has become a powerful tool in exploring and identifying patterns in genome data.

A vast number of clustering algorithms have been developed in the literature. The model-based methods (Banfield and Raftery, 1993) make an assumption on the probabilistic distribution of data, and the distance-based methods employ the notion of "distance" that represents the similarity between two data points. Among the distance-based methods, two major types are most commonly used. The first type is hierarchical clustering (Hartigan, 1975), which partitions data into groups through a series of agglomerative or divisive steps that operate on the similarity measure between data points. The structure of data is revealed through the process of hierarchical clustering and is presented by a tree diagram known as dendrogram. One weakness of hierarchical clustering is the irrevocable clustering assignments: A mistake made at early steps can never be corrected at later steps.

The second type of distance-based methods is known as the partition clustering. The clustering results are obtained as an optimal solution that either maximizes or minimizes a criterion of some kind. The k-means algorithm (MacQueen, 1967; Lloyd, 1982) that adopts the criterion of minimizing the sum of squared distances from each data point to its closest cluster center is one clustering algorithm of this sort. Such algorithms, however, usually require an initial partition to start the iterative process, and the number of clusters has to be given a priori. In addition, this type of algorithms suffers from the problem of trapping into local minima (or maxima), which is a result of a poor selection of the initial partition. There exist methods to improve the performance of k-means algorithm, including the estimation of the number of clusters (Milligan and Cooper, 1985; Tibshirani et al., 2001) and the selection of initial values to solve the local minima problem (Selim and Alsultan, 1991; Tseng and Wong, 2005).

Despite its few weaknesses, k-means algorithm is by far the most popular clustering algorithm used in research and industrial applications. The speed and the simplicity make k-means algorithm appealing. However, it does not guarantee a better performance compared to other clustering algorithms. In addition to the selection of an initial partition that may result in a poor performance, the criterion used by k-means to define the "optimal" groupings sometimes is inappropriate. Section 4 shows an example in which the structure of data does not conform to this criterion.

So it comes to the major challenge that every clustering problem encounters: How to define an appropriate criterion for clustering? Is there a criterion that makes sense to all data? We may answer this question by asking "what does cluster mean?" The word, cluster, means (1) a group of the same kind, or (2) a group of things close together. These explanations describe the nature of clustering, and are indeed how human perception identifies clusters: elements that are close or resembled belong to the same group.

The formulation of a criterion, such as minimizing the sum of within group variations by k-means, is a conceptually and technically useful approach to the problem of clustering. However, it does not pertain to the nature of clustering as described above. From this point of view, hierarchical clustering is favored by us, because it employs the idea of closeness and resemblance between elements. One weakness of hierarchical clustering is that elements are merged too fast.

We will show in Section 4 that our algorithm slows down the merging process and is benefited from it.

In addition to defining a clustering criterion, another challenge for clustering algorithms is when the amount of noise in the data is substantial. Such situations arise very often in the analysis of gene expression data, in which a significant number of observed genes have distinct expression profiles and do not co-regulate with other genes. The high level of noise from these so called "scatter noise genes" is likely to obscure or even distort a meaningful underlying expression pattern. The identification of such genes is therefore crucial to the performance of a clustering method. Recently several proposals have been made for clustering data with noise set. Some methods allow noise to remain unclustered (Fraley and Raftery, 2002; Tseng and Wong, 2005). Some incorporate functional annotation of genes into the analysis of microarray data (Pan, 2006; Shen et al., 2010)

The self-updating process is a clustering algorithm that overcomes the aforementioned problems and challenge for clustering. The development of this algorithm was initiated as an extension to the generalized association plots (GAP) (Chen, 2002), which was at first a clustering method and was later integrated into a platform for exploratory data analysis (Wu et al., 2010). GAP utilizes the iterative generated correlation matrices (Mcquitty, 1968), according to which data points are gathered towards the left and the right sides of an ellipse at each iteration and eventually merge into two clusters. To extend GAP, we first introduced a threshold on correlation to increase the number of clusters the iterative matrices converge to. We later moved the iteration process from the correlation space to the raw data space, that can display the actual movements of data points. In the resulting clustering process, each data point continues updating its own location until the whole system reaches a balance condition. We therefore named this algorithm Self-Updating Process (SUP).

Since the beginning of this work in the year of 2006, we considered the idea of SUP original and new. Not until recently have we become aware that this idea has been introduced into the literature as the blurring mean-shift algorithm (Fukunaga and Hostetler, 1975; Cheng, 1995). In contrast to SUP that was exclusively developed for the problem of clustering, the original mean-shift algorithm (Fukunaga and Hostetler, 1975) made its first appearance for kernel density estimation, which uses the sample mean within a local region to estimate the gradient of a density function. The mean-shift algorithm was further extended and analyzed by Cheng (1995), in which a generalized version called "blurring" mean-shift is equivalent to SUP. Comaniciu and Meer (2002) applied the mean-shift algorithm to the problem of image segmentation. The algorithm has become more well-known in the computer science community since then.

While we independently came up with the same idea as the blurring mean-shift algorithm, we also made great efforts to develop the updating system and to study its properties. For example, we defined parameters to have specific statistical meanings, in order to better reflect the intuition behind the idea. We studied the influence of parameters on clustering results, and we provided a guideline

for parameter estimation. We made a thorough study on the performance of this algorithm, and we conducted simulation experiments to demonstrate its strengths. We also discovered a distinctive characteristic of this algorithm: it has the ability to simultaneously isolate noise while performing clustering. This characteristic makes the algorithm particularly powerful for data that contains a significant amount of noise. We present this characteristic by a simulation example in Section 4.1 and by a gene expression data in Section 5.1. All of the aforementioned work, to our knowledge, has not yet been reported into the literature.

In this paper we present a compete development of SUP. The paper is organized as follows. Section 2 introduces the clustering algorithm SUP. In Section 3, we provide a mathematical proof that guarantees the convergence of SUP. Section 4 presents simulations that demonstrate the performance of SUP and shows comparisons with other methods. Illustrative examples are given in Section 5, including a clustering analysis of gene expression data (Golub et al., 1999) and an application to image segmentation. A discussion is presented in Section 6.

2 Self-Updating Process

2.1 The idea

The central idea of self-updating process can be illustrated by the following example.

Recall times when we were students and our teacher asked us to divide into groups to play a game. What did we do? We might walk directly towards our good friends. We might even ask some of those next to us whether they wanted to be in the same group while we walked. On the teacher's side, he/she would see that students are moving. Gradually and eventually, the groups were formed.

Based on this childhood experience, we describe our algorithm as follows. Suppose there are N elements to be clustered, and there are p random variables representing elements' information. The data is a $N \times p$ matrix, which can be viewed as N data points in a p-dimensional space. When the updating process begins, the movement of a data point is determined by its relationship with other data points. The relationship, for example, can be friendships and locations as in the previous example. We can quantify the relationships based on elements' information using measures such as the correlation and the Euclidean distance.

2.2 The main algorithm

The self-updating process is formulated as follows.

(i) $x_1^{(0)}, x_2^{(0)}, \cdots, x_N^{(0)} \in R^p$ are data points to be clustered.

(ii) At time $t+1$, every point is updated to

$$x_i^{(t+1)} = \sum_{j=1}^{N} \frac{f(x_i^{(t)}, x_j^{(t)})}{\sum_{k=1}^{N} f(x_i^{(t)}, x_k^{(t)})} x_j^{(t)}, \tag{1}$$

where $f(x_i^{(t)}, x_j^{(t)})$ is some function that measures the influence between $x_i^{(t)}$ and $x_j^{(t)}$.

(iii) Repeat (ii) until every point converges.

When two data points are closer, the influence between them should be stronger. We therefore assign a larger value to f when $x_i^{(t)}$ and $x_j^{(t)}$ are closer, and interpret $f(x_i^{(t)}, x_j^{(t)})$ as the mutual influence between point i and point j at the t-th update. In plain words, equation (1) says that the next location where point i moves to is determined by influences it receives from all data points at present, including point i itself. In statistical terminology, $x_i^{(t+1)}$ is *the weighted average* of all $x_i^{(t)}$'s, for $i \in \{1, ..., N\}$.

Throughout this paper we use f as an exponential decay function with respect to some distance d:

$$\begin{aligned} & f(x_i^{(t)}, x_j^{(t)}) \\ = & \begin{cases} \exp[-d(x_i^{(t)}, x_j^{(t)})/T], & d(x_i^{(t)}, x_j^{(t)}) \le r \\ 0, & d(x_i^{(t)}, x_j^{(t)}) > r, \end{cases} \end{aligned} \tag{2}$$

where we select $d(x_i^{(t)}, x_j^{(t)})$ as the Euclidean distance between locations of point i and point j at the t-th update. In the next subsection we use a simple example to demonstrate how SUP operates. We also illustrate the roles of r and T in the example. Other formulations of f's are discussed in the final section.

Note that in the original mean-shift algorithm (Fukunaga and Hostetler, 1975), f in (1) is defined as a flat kernel, which is an indicator function representing whether the distance between two points is less than some threshold value. Cheng (1995) further generalized f to be any kernel function, while most of the mean shift applications used a truncated Gaussian as the kernel. In this paper we use a truncated Exponential function as written in (2), because exponential decay is very often observed in nature. Moreover, in the later discussion we show that the parameter T can be "dynamic": the parameter value decreases with iterations. That is to say, the updating process is no longer homogeneous. This is one case not considered in the mean shift algorithm.

2.3 A simple illustration

Three data points from bivariate normal distributions $BVN(\mu_k, I_2/25)$ were sampled for each $k \in \{1, ..., 9\}$, where $\mu_k \in \{$(0,0), (2,0), (1,1), (6,0), (8,0), (7,1), (3,3), (5,3) and (4,4)$\}$. A total of 27 sampled points were plotted in

Figure 1(a). We first used the f in (2) with $r = 0.9$ and $T = 0.7$. Figures 1(a)-(d) illustrate how SUP updated the location of each data point. At $t = 3$, the 27 data points converged to nine points without further movements afterwards. These nine points represent final locations of the resulting nine clusters.

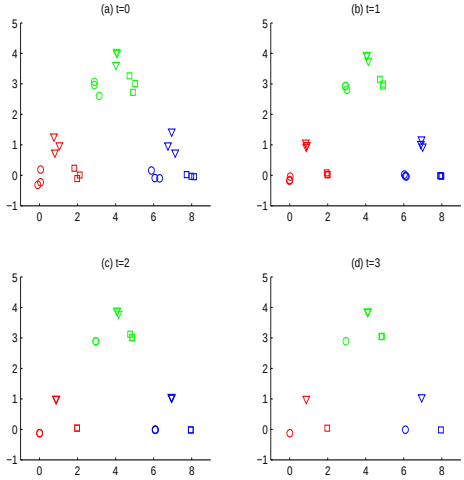

Figure 1: A graphical presentation of SUP with r=0.9 and T=0.7

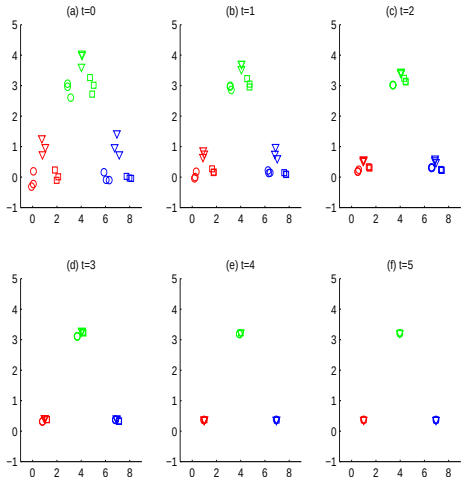

Figure 2: A graphical presentation of SUP with r=3.5 and T=0.7

To illustrate the effect of r, we increased r from 0.9 to 3.5. Figure 2(a)-(f) present the updating process and the final clustering result, in which data points converged to three locations instead of nine. Indeed, when we take a look at the original sampled data in Figure 1(a), it is subjective to conclude whether there

are three or nine groups. This is the role we assign r to play, that defines *the range of influence*. In this example, a choice of $r = 3.5$ forced each data point only to be influenced by those who were within 3.5 units. As a result, squares, circles and triangles of the same colors were jointly influenced, and in the end of the process converged to the same locations. Generally speaking, the use of a small r value produces clusters of compact sizes, within which data points are less heterogeneous. Without the use of r, or equivalently, when r is infinite, Corollary 1 in the next section proves that all data points eventually converge to a single cluster for any strictly positive f function.

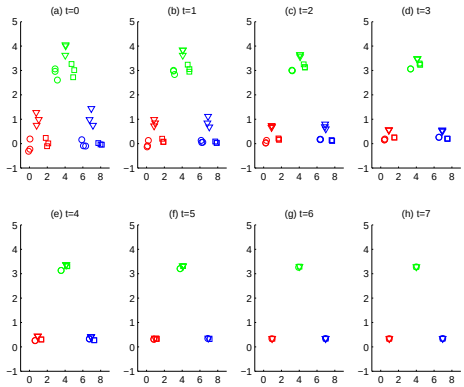

Figure 3: A graphical presentation of SUP with r=3.5 and T=0.5

To illustrate the effect of T, we changed T from 0.7 to 0.5, keeping r at 3.5. Figure 3(a)-(h) present the updating process and the final clustering result. A comparison between Figure 2 and Figure 3 shows that data points converged at a slower rate when T is smaller. This can also be seen from (2): When T is small, $f(u, u) = 1$ is much larger than $f(u, v)$ for every $v \neq u$. Data point u therefore hardly moves as the influence from itself totally dominates; When T is large, similarly we can conclude that the movements of data points are correspondingly large. If we consider data points as particles in a statistical mechanical system, parameter T then serves as the "temperature". Data points move fast at a high temperature T, therefore accelerate the convergence speed. That is to say, parameter T determines the rate of convergence of SUP. Different T values may produce different clustering results, which are due to different merging speeds.

2.4 Parameter Estimation

In self-updating process, there are two parameters to be determined. One is r and the other is T. If there is a training set, Cross-Validation is a standard method to estimate the parameters. However, in practice we rarely have this additional information to learn the parameter values. In the following we present simple data-driven methods for parameter estimation.

2.4.1 The influential range r

The selection of the influential range r can depend on the estimated probability density function of the pairwise distance. The valleys and peaks of the density function provide useful information about good and poor candidates of r. The reasoning is as follows.

We begin with a simple situation when there are only two clusters in the data. Because of the two-cluster structure, the pairwise distances of pairs that contain one point from each cluster should not differ much, meaning that the estimated probability density function of the pairwise distance has a large probability mass in the range of these between-cluster distances. Similarly, the pairwise distances of pairs that contain both points from the same cluster should not differ much. There should also be a large probability mass in the range of these within-cluster distances. To select an influential range r that produces clusters retaining the original structure of two clusters, we should avoid these distances at peaks that are likely to be the between- or within-cluster distances. Otherwise, the updating process may easily distort the original structure of data and may consequently result in a poor clustering performance.

The same reasoning applies to data of more than two clusters. Following this reasoning, a good candidate of r is either larger or smaller than the distances at peaks. A more desirable choice is a distance at valley with a small probability mass. This valley selection minimizes the chance that the updating process distorts the data structure, because the number of pairs that are influenced by the value of r is small compared to other choices of r values.

We take the data presented in Figure 1(a) as an example. We use frequency polygon to approximate the probability density function of the pairwise distance. Figure 4 presents the frequency polygon, in which valleys are at around 0.9, 2.5, 3.5, 5.1, 6.8 and 7.7. We showed in Section 2.3 that the use of $r = 0.9$ produced three clusters and that of $r = 3.5$ produced nine clusters. For the rest of the valleys, $r = 2.5$ produced an identical clustering result as $r = 3.5$, while $r = 5.1$, 6.8 and 7.7 moved all data points into one single cluster.

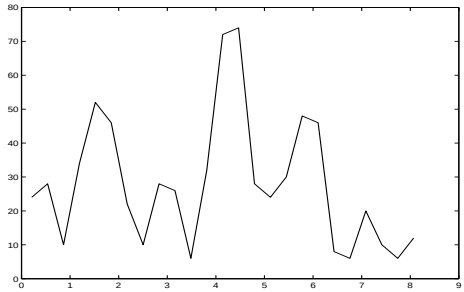

Figure 4: The frequency polygon of the pairwise distances.

2.4.2 The temperature T

As we learned in Section 2.3 that the temperature T determines the convergence rate of SUP, one may favor a large T value to reduce the computation time. It is, however, not usually the case. When data points move fast at a high temperature, mistakes are often made.

To select a temperature T, we consider the following. Let data points j and k be $r - \delta$ and $r + \delta$ units away from data point i, respectively, where r is the selected influential range. When δ is small, i is about r units away from both j and k. It is reasonable to assume that i receives approximately the same amount of influence from j and k. According to (2), however, the actual influences that i receives from j and k are $\exp[-(r+\delta)/T]$ and zero, respectively. It therefore makes more sense to use a small T value such that $\exp(-r/T)$ is close to zero. When $T = r/5$, $\exp(-r/T)$ is 0.0063. Our experiments showed that SUP with $T = r/5$ very often produced good clustering results within a reasonable computing time.

An alternative choice to a static temperature of $T = r/5$ is a dynamic temperature that increases with time. In the beginning of the updating process, data points move at a lower temperature to slow down the merging. The temperature is gradually increased with time to accelerate the updating process. For dynamic temperature, we propose to use $T = r(1/20 + t/50)$. The initial temperature is only $r/20$. After $t = 8$, the temperature exceeds $r/5$. From our experiments, the use of a dynamic temperature is usually capable to handle a wider variety of data.

3 Convergence

In this section we prove the convergence of the self-updating process.

The convergence of blurring mean-shift algorithm was studied by Cheng (1995), which however, included only two cases. The first case was when the mutual influence between each pair of elements is nonzero. Theorem 3 in Cheng (1995) showed that in this case all elements eventually converge to a single cluster. The second case is under the assumption that elements can never move arbitrarily close to each other. Theorem 4 in Cheng (1995) guaranteed that in this second case the algorithm converges in finite steps.

Our result is more general than Theorem 3 and Theorem 4 in Cheng (1995), where the Theorem 3 is equivalent to our Corollary 1, which is an immediate implication of our main result presented in Theorem 1. The convergence of self-updating process requires the function f to be *PDD* (positive and decreasing with respect to distance), that we define in the following.

Definition 1. *The function f in (1) is PDD (positive and decreasing with respect to distance), if*

(i) $0 \le f(u, v) \le 1$, *and* $f(u, v) = 1$ *only when* $u = v$.

(ii) $f(u, v)$ *depends only on* $\|u - v\|$, *the distance from* u *to* v.

(iii) $f(u, v)$ is decreasing with respect to $\|u - v\|$,

Since $f(u, v)$ represents the influence that data points u and v receive from each other, we assume a larger value of $f(u, v)$ when u and v are closer, as stated in condition (iii). We also assume that the influence is solely determined by the distance between u and v, meaning that $f(u_1, v_1) = f(u_2, v_2)$ whenever $\|u_1 - v_1\| = \|u_2 - v_2\|$, as stated in condition (ii). In principle, $f(u, v)$ can be negative, suggesting that u and v repel each other. However, when $f(u, v)$ is negative at some iteration and since f is decreasing, u and v would move further and further apart as the process keeps updating. The whole system will consequently diverge and never reach a balance condition. This is the reason we require $f(u, v)$ to be non-negative. We can define $f(u, u)$ to be any positive number. For simplicity we normalize it to be one.

With a function f that satisfies *PDD* condition, the following theorem guarantees the convergence of SUP.

Theorem 1. *If the function f in (1) is PDD, there exists $\{x_1, x_2, \cdots, x_N\}$, such that*

$$\lim_{t\to\infty} x_i^{(t)} = x_i \qquad \forall i.$$

To prove theorem 1, we introduce lemma 1, lemma 2 and lemma 3.

Lemma 1. *Let $C_1^{(t)}$ be the convex hull of $\{x_1^{(t)}, x_2^{(t)}, \cdots, x_N^{(t)}\}$. Then*

$$C_1^{(0)} \supseteq C_1^{(1)} \supseteq \cdots \supseteq C_1^{(t)} \supseteq \cdots .$$

Proof. The convex hull $C(X)$ for a set of points X is the minimal convex set containing X. Since

$$x_i^{(t+1)} = \frac{\sum_{j=1}^{N} f(x_i^{(t)}, x_j^{(t)}) \cdot x_j^{(t)}}{\sum_{j=1}^{N} f(x_i^{(t)}, x_j^{(t)})},$$

$x_i^{(t+1)}$ is a weighted average of $x_j^{(t)}$ for $j = 1, \cdots, N$. Therefore,

$$x_i^{(t+1)} \in C_1^{(t)}.$$

Since the above is true for each i, we have

$$C_1^{(t)} \supseteq C(\{x_1^{(t+1)}, x_2^{(t+1)}, \cdots, x_N^{(t+1)}\}) = C_1^{(t+1)}.$$

□ □

Note that the nested structure presented in lemma 1 ensures the convergence of convex hulls $\{C_1^{(t)}\}$. Let C_1 be the limit of $C_1^{(t)}$,

$$C_1 \equiv \lim_{t\to\infty} C_1^{(t)} = \bigcap_{t=0}^{\infty} C_1^{(t)}.$$

On the other hand, since the convex hull of any finite set of points in R^p is a polytope, each $C_1^{(t)}$ is a polytope. Each vertex of $C_1^{(t)}$ therefore must contain at least one $x_i^{(t)}$ for some i, otherwise the polytope would have been smaller. With the convergence of convex hulls $\{C_1^{(t)}\}$, lemma 2 claims that there are at least some data points $\{x_i^{(t)}\}$ which converge to vertices of C_1. The proof is included in Appendix A.1.

Lemma 2. *If the function f in (1) is PDD, for each vertex $v_{1,i}$ of C_1, there exists at least one j, such that*

$$\lim_{t\to\infty} x_j^{(t)} = v_{1,i}. \tag{3}$$

Having shown that at least some points converge under SUP, hereafter we consider the rest of the data points. Let Ω_1 be the set of points shown converged to the vertices of C_1. Define $C_2^{(t)}$ be the convex hull of $\{x_i^{(t)}\}_{i\notin\Omega_1}$. Note that $\{C_2^{(t)}\}$ may not be nested at early stages of iterations: points not in Ω_1 may move outside the current convex hull $C_2^{(t)}$ due to the influence from Ω_1, the volume of the convex hull therefore may increase by iteration. This nested property, however, would hold after some iteration when all data points in Ω_1 converge. Explicitly,

$$C_2^{(t)} \supseteq C_2^{(t+1)} \text{ after some t,}$$

which also implies the convergence of $\{C_2^{(t)}\}$,

$$C_2 \equiv \lim_{t\to\infty} C_2^{(t)}.$$

We introduce the following lemma 3, which can lead to the nested property of $\{C_2^{(t)}\}$. It states that when all data points in Ω_1 converge, points in Ω_1 receive no influence from points not in Ω_1, otherwise they would have been attracted inwards. That is to say, data points not in Ω_1 also no longer receive influence from points in Ω_1, meaning that the influence from points in Ω_1 goes down to zero. The proof of lemma 3 is included in Appendix A.2.

Lemma 3. *For any x_i such that $i \in \Omega_1$,*

$$\lim_{t\to\infty} f(x_i^{(t)}, x_j^{(t)}) = 0,$$

for all j such that

$$\lim_{t\to\infty} x_j^{(t)} \neq \lim_{t\to\infty} x_i^{(t)}.$$

From the above, we can claim a similar result for C_2 as lemma 2 for C_1: each of the vertex of C_2 has at least one data point converges to. The same argument can apply again and again to C_3, C_4, $\cdots$, until all data points converge. This completes the proof of theorem 1.

Although theorem 1 guarantees the convergence of SUP when f has *PDD* condition, there are some f's that produce trivial clustering results, in which all data points are clustered into one single group. We identify such f's in the following corollary.

Corollary 1. *Let r_M be the maximum pairwise distance between any two data points. If f is PDD with $f(r_M) > 0$, there exists c, such that*

$$\lim_{t\to\infty} x_i^{(t)} = c \qquad \forall i.$$

For any two points i and j, lemma 1 implies that $||x_i^{(t)} - x_j^{(t)}|| \leq r_M$ for every t. Since f is decreasing with respect to distance, the influence between i and j is always larger than $f(r_M)$. If $f(r_M) > 0$, then $f(x_i^{(t)}, x_j^{(t)}) \geq f(r_M) > 0$ for every i and j. Lemma 3 shows that, however, the influence between any two points which do not converge to the same position tends to zero. Thus, $f(x_i^{(t)}, x_j^{(t)}) \geq f(r_M) > 0$ for every i and j implies that all data points converge to the same position, as stated in corollary 1.

For the purpose of clustering, it is not desirable to have all data points converged to the same position. To prevent trivial clustering results, f has to be zero on (r, ∞) for some $r < r_M$.

4 Simulations and Comparison

4.1 Data with noise

In this simulation we consider data that contains noise, which in practice is very often present, for example, in gene expression data and in image data. Many clustering algorithms sometimes fail to produce reasonable results for data with noise, because scattered points of noise very often obscure the original structure of data, therefore make it difficult for algorithms to discover patterns.

We demonstrate the performance of SUP in comparison with k-means algorithm, considering two approaches to solve the problem of local minimum that may result in a poor performance of k-means. The first approach was the use of multiple initial values. We allowed k-means to start with multiple sets of k randomly selected initial centers. Then the clustering result from the set of initial centers that had the minimum sum of within group variations was selected. It is clear that the use of multiple initial values can increase the chance for k-means to achieve its optimal performance. The second approach we considered was the selection of initial values by Tseng and Wong (2005), in which the hierarchical clustering was first applied to obtain results of $k \times p$ clusters for some pre-determined number of clusters k and some integer value p. Then the centers of the k largest clusters were taken as the initial centers for k-means. Tseng and Wong (2005) showed that this method for selecting initial centers can achieve good clustering results.

We used the example also presented by Tseng and Wong (2005), in which data of three clusters and a number of scattered points were generated as follows. Three clusters were sampled from standard normal distributions centered at (-6,0), (6,0) and (0 6), respectively. Each point in the cluster was restricted to lie within two standard deviations to its center. The scattered points representing noise were sampled uniformly from [-12, 12] $\times$ [-6, 12], but not within three

standard deviations to any of the three centers. We used different numbers of scattered points to represent data sets with varying degrees of noise: each simulated data has 50 points in each of the three clusters and n scattered points, where n can be 10, 50, 100 or 200. Figure 5 displays one example data, showing 50 points in each of the three clusters denoted by circles, x-marks and pluses, and 50 scattered points of noise by dots.

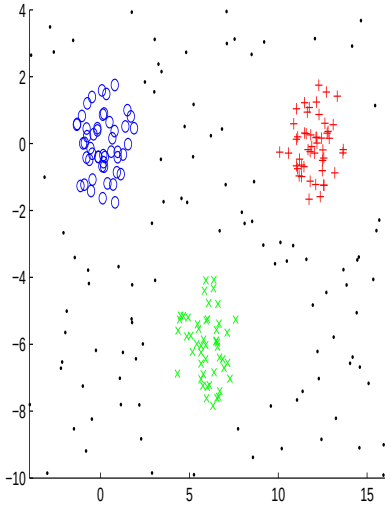

Figure 5: Three groups (circles, x-marks and pluses) and noises (dots)

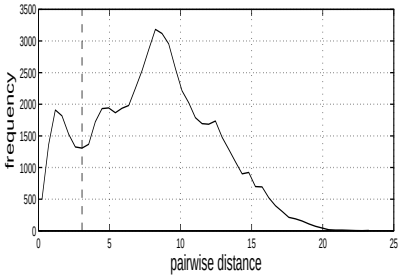

Figure 6: The frequency polygon of the pairwise distances with one valley at around 3.0618

We simulated 100, 000 runs for each data size, and compared clustering results from k-means algorithm and those from SUP. For k-means algorithm, results were obtained using random initials of one set, random initials of 100 sets, and initials by Tseng and Wong (2005) using single and complete linkages with

Table 1: The numbers of incorrect results in 100,000 runs of simulations

Number of noise			10	50	100	200
K-means	random initial		9774	1177	1150	1404
	100 sets of random initials		0	0	0	709
	Single Linkage	p=1	6771	1471	1143	1153
		p=3	43	3085	2211	1114
		p=6	0	551	925	963
	Complete Linkage	p=1	1849	1380	952	1251
		p=3	9	0	0	444
		p=6	4597	1	1	337
SUP	static T		0	0	0	16
	dynamic T		0	0	0	205

Table 2: CPU time per run

Number of noise		10	50	100	200
K-means	random initial	0.0026	0.0031	0.0036	0.0037
	100 sets of random initials	0.1487	0.1712	0.2233	0.2463
SUP	static T	0.0197	0.0558	0.0922	0.2481
	dynamic T	0.0255	0.0532	0.0811	0.1988

p=1, 3, and 6. For SUP, we present results from both static and dynamic temperatures with $T = r/5$ and $T = r(1/20 + t/50)$, respectively, where the value of r was selected automatically at a time for each simulated data according to the frequency polygon of the pairwise distances. Figure 6 graphically shows the selection of r for the example data presented in Figure 5. The frequency polygon suggests the use of $r = 3.0618$.

While the objective is to correctly cluster the 150 non-noise points, we compared SUP and k-means algorithm on the basis of the number of runs that produced correct clustering results: a run was taken as "correct" only when all of the 150 non-noise points were clustered correctly. Table 1 presents the number of runs that were not correct for data containing different levels of noise. This table demonstrates the clustering performance of SUP, showing that SUP made considerably fewer mistakes than k-means algorithm. We also computed the running time in seconds for each run of the simulation. Table 2 focuses on the comparison between SUP and k-means algorithm with multiple initial sets, which also produced reasonable clustering performance as shown in Table 1. This comparison suggests that SUP is competitive in computation efficiency.

In addition to the 150 non-noise points, we examined clustering results of the scattered noise data points. We take the example presented in Figure 5 to illustrate the results. When $r = 3.0618$ was used, SUP produced 12 clusters. Three, four and five noise data points were grouped into the three clusters of 50

non-noise points, respectively. The rest of the 38 noise data points constituted the other nine clusters. When r decreased to 2.1, SUP produced 26 clusters, including three clusters of 50 non-noise data points and 23 clusters constituted exclusively by the 150 scattered noise data points. When r further decreased to 1.5, SUP produced 34 clusters. The three clusters of 50 non-noise data points remained, and the number of clusters constituted by scattered noise data points increased to 31.

To summarize, this simulation example shows that SUP has superior performance over k-means algorithm in clustering data with noise. In addition, SUP has the ability to separate the noise data points from the non-noise data points. This ability is further demonstrated by heat maps presented in Figure 10 based on a gene expression data.

4.2 Crowded Data

When groups in the data are widely separated from each other, most of the clustering algorithms can produce good results. In this simulation example we focus on the type of data in which groups are closely separated. Fifty points were sampled for each of the three groups from bivariate normal distributions centered at $(-4, 5)$, (-5, -1) and $(0, 1)$ with $(\sigma_x, \sigma_y, \rho)$ as $(5, 2, 0)$, $(2, 2, 0)$ and $(3, 3, 0)$, respectively. Each point was restricted to lie within one standard deviation from its center. The data points in the first group was further rotated counterclockwise with respect to its center $(-4, 5)$ to create a structure of an inclined ellipse. Figure 7(a) displays an example of simulated data, in which points from the three groups were colored in navy, red and green, respectively.

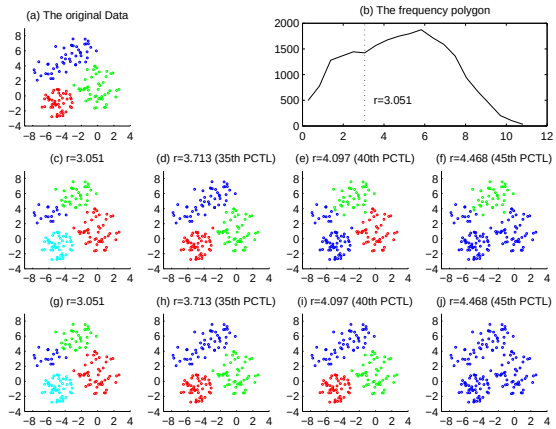

Figure 7: (a) The example data. (b) The selection of r value. (c)-(j) Clustering results by SUP using different r values, where (c)-(f) are results from static temperature and (g)-(j) are from dynamic temperature. Data points that were clustered into the same group are displayed by the same color.

Since groups in this example were closely located, sometimes it is difficult to find a local minimum of pairwise distances for this type of data. Figure 7(b) shows the frequency polygon of the pairwise distances for the example data presented in Figure 7(a). There was an unclear valley at around 3.051. This r value produced four clusters, presented in Figures 7(c) and 7(g) that are results from static and dynamic temperatures, respectively. To enlarge the cluster sizes, we increased the value of r by experimenting with various percentiles of the observed pairwise distances. Figures 7(d)-(f) and Figures 7(h)-(j) present results from taking r as the $35th$, $40th$, and $45th$ percentiles. Figures 7(d)-(f) show results from static temperature and Figures 7(h)-(j) show results from dynamic temperature.

It is noticeable that the use of dynamic temperature outperforms that of static temperature in this simulation example. The detailed description is as follows. Although Figure 7(c) shows that a correct three-cluster result can be obtained by merging the navies and the greens, Figures 7(d)-(f) show that the use of static temperature with other values of r either made one mistake or produced clustering results that displayed wrong data structure from the original data. Figures 7(h) and 7(i), on the other hand, show that SUP with dynamic temperature using r as the $35th$ and $40th$ percentiles both produced perfect clustering results. When r increased to the $45th$ percentile, meaning that every point was influenced by almost half of the points in the data, the updating process assigned all data points to a single cluster, as shown in Figure 7(j).

When groups in the data are closely apart as in this simulation example, data points should be updated much slower especially at early iterations of the process. This is the case when we use the dynamic temperature $T = r(1/20+t/50)$, where the initial temperature is only $T = r/20$, a considerably low temperature compared to the static temperature $T = r/5$. From this point of view, the use of dynamic temperature is more appropriate for clustering crowded data.

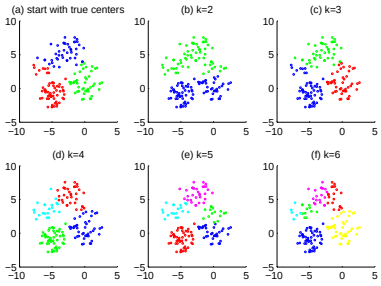

Figure 8: Clustering results by k-means algorithm. (a) is the result using the true centers as initials, and (b)-(f) are results from random initials.

We also applied k-means algorithm and hierarchical clustering to the same data presented in Figure 7(a). Figures 8(a)-(f) show results from k-means al-

gorithm, where Figure 8(a) was the result using the true centers $(-4, 5)$, (-5, -1) and $(0, 1)$ as initials, and Figures 8(b)-(f) were results from random initials with varying k's, where k is the pre-determined number of groups. These figures show that k-means was unable to capture the original structure of the data, even when we used the true centers as the initial values.

While k-means is designed to minimize the total distances between each data point to its cluster centers, it may mistakenly assign data points that belong to a larger cluster to a nearby smaller one, as these points are in fact closer to the center of the smaller cluster. This explains what we see in Figures 8(a)-(f), that k-means is likely to fail when data has clusters of distinct shapes and sizes as in this simulation example.

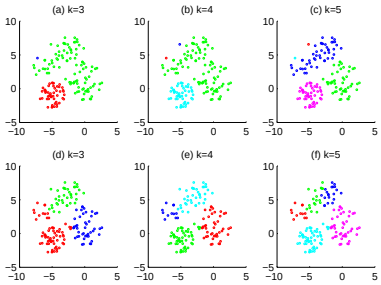

Figure 9: Hierarchical clustering results. (a)-(c) are results by single linkage, and (d)-(f) are results by complete linkage.

The results by hierarchical clustering using single and complete linkage are presented in Figures 9(a)-(c) and Figures 9(d)-(f), respectively. The results by single linkage agree more to the clustering structure of the original data. However, as it is often difficult to find a good threshold for cutting the dendrogram tree into a clean and meaningful clustering structure, some subsequent merging and rotation of the sub-trees are often necessary.

5 Application

5.1 Golub Data

We use the gene expression data presented in Golub et al. (1999) to demonstrate the clustering performance of SUP and its ability to separate noise. Three preprocessing steps were applied to Golub data that originally has expression values of 7129 genes from 38 patients. Normalizations of the expression values within arrays were also applied. The data we obtained was from the package "multtest" (version 2.8.0) of Bioconductor. This data contains pre-processed and normalized expression values of 3105 genes from 38 patients, among which 27 were with acute lymphoblastic leukemia (ALL) and 11 were with acute myeloid leukemia (AML). The following illustration of SUP includes: (i) discover gene patterns

that are mostly associated with ALL-AML distinction, and (ii) classify the 38 patients in this sample data using the 50 genes selected by Golub et al. (1999).

5.1.1 Discover gene patterns

Recall that SUP updates each data point's location according to the function f that measures the influence between every pair of points. The more similarity between two points, the more influence the two points receive from each other. From this point of view, the resulting clusters that have only one data point or two are considered isolated, showing no resemblance to the rest of the data. These isolated data points we often call "noise". In the context of gene expression data, we consider these isolated data points as scattered genes.

To perform SUP on genes, we first normalized the expression values by genes to ensure equal weight of each gene. Then we calculated the frequency polygon of the pairwise distances between genes to determine the influential range r. In the frequency polygon, there showed no clear valley. We then found that the use of $r = 5$ produced only one large cluster and many tiny clusters that were considered as noise. Figure 10 presents clustering results by SUP using dynamic temperature with selected r values smaller than 5. It is easily noted that smaller r values produced tighter clusters that exhibited lower within-cluster heterogeneity.

We summarize the clustering results by taking $r = 4.6$ as an example. The use of $r = 4.6$ produced 1478 clusters in total, among which there were only nine clusters of sizes larger than ten, and 1420 clusters that contained only one single gene. The sizes of the five largest clusters were 580, 349, 276, 176 and 38 genes, respectively. The largest and the fourth largest clusters, as shown in Figure 10(c), corresponded to genes that had expression values above the mean (colored in red) for most of the ALL patients and below the mean (colored in blue) for most of the AML patients. The distinction between the first and the fourth largest clusters was the detailed expression patterns shown particularly in ALL patients. The second and the third largest clusters, in contrast, corresponded to genes that had high expression values (colored in red) for most of the AML patients. The distinction between the second and the third largest clusters was also the expression patterns shown in ALL patients: the third largest cluster exhibited uniformly low gene expression values, while the genes in the second largest cluster had both high and low values.

To validate the performance of SUP, we compared our clustering results to the 50 genes identified by Golub et al. (1999) that were most highly correlated with ALL-AML distinction. We located the 50 genes in the clusters produced by $r = 4.6$, finding that 25 of them were included in the largest cluster, 24 were in the third largest cluster and one gene in the second largest cluster. When $r = 4.7$ was used, all of the 50 genes were found in the two largest clusters, with 25 genes in each of the two. Figure 10(d) shows that these two largest clusters corresponded to genes with high values in AML patients and genes with low values in AML patients, respectively.

In this application, we also demonstrate the strength of SUP in isolating

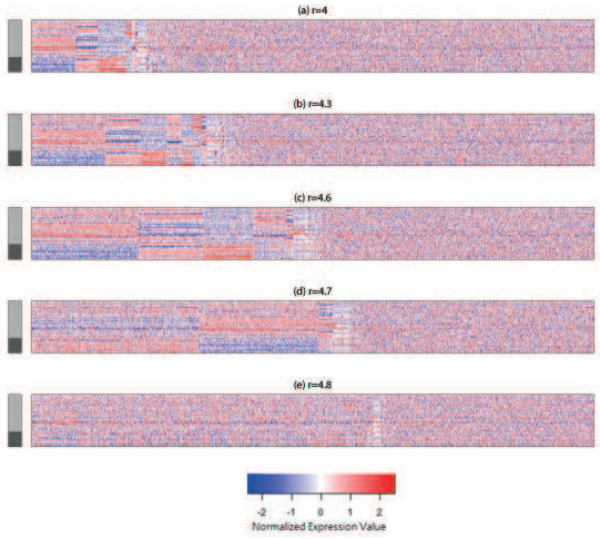

Figure 10: The clustering results by SUP with dynamic temperature and various values of r. The side-bar next to each of the heat-map indicates patients' cancer types, where light and dark gray colors represent ALL and AML patients, respectively. The heat-maps display normalized gene expression values of 3051 genes from 38 patients, with each row representing a patient's gene expression profile.

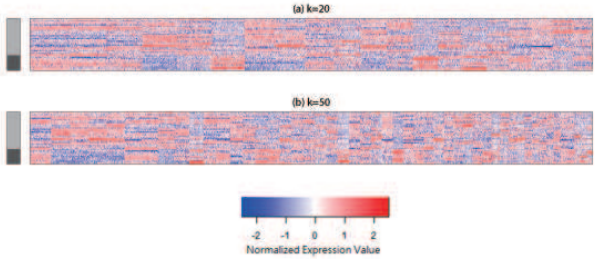

Figure 11: The clustering results by k-means algorithm with different values of k's.

noise by a comparison to k-means algorithm. Figure 11 presents clustering results by k-means algorithm with $k = 20$ and $k = 50$. The clusters produced by k-means were of similar sizes, and the 50 significant genes identified by Golub et al. (1999) dispersed to nine and twelve clusters when $k = 20$ and $k = 50$, respectively. As a result, meaningful gene expression patterns were difficult to obtain by k-means algorithm, unless we had a way to merge and clean the clustering results.

While Figure 11 shows that k-means algorithm is incapable of separating scattered genes, this lack of ability to separate noise is in fact a common problem of most of the clustering algorithms, including hierarchical clustering.

5.1.2 Classify patients

We performed SUP on the 38 patients in this data, using expression values of the 50 genes that were most correlated with ALL-AML distinction identified by Golub et al. (1999). Since the 50 genes were selected using exactly the same data from the 38 patients we want to classify, this clustering analysis basically has no practical value. The purpose of this analysis, however, is simply to compare the clustering performance of SUP to that of k-means and hierarchical clustering, taking this gene expression data of size 38×50 as an illustrative example.

The pairwise distances between patients were calculated. Figure 12 shows the frequency polygon, in which a clear valley was observed at around 9.8982. Using this r value, SUP with dynamic temperature produced two clusters of sizes 27 and 11. These two clusters identically corresponded to the two groups of patients, ALL and AML, meaning that SUP made 100% accurate distinction between the two types of leukemia. We also applied k-means algorithm and hierarchical clustering to this data. Results showed that k-means algorithm with $k = 2$ misclassified one patient, and hierarchical clustering with different linkages misclassified at least two patients.

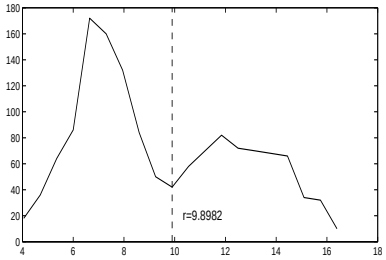

Figure 12: The frequency polygon of pairwise distances between patients

5.2 Image Segmentation

When pixels in an image are considered as elements to be clustered, the problem of image segmentation becomes a clustering problem. In this example we demonstrate the performance of SUP on the problem of image segmentation. We selected six test images from "The Berkeley Segmentation Dataset". Figure 13 displays the test images. Each image is of size 240×160.

The first question we encountered was: what information should be included in the data, that can best convey important characteristics of an image for the

purpose of segmentation? We consider two types of information. One is the position, and the other is the color intensity. The information on position is represented by two variables: the x and y coordinates of a pixel. The information on color intensity can be obtained through the components of a color model. Among several choices of the representation for color intensity, the YUV model that defines a color space in terms of one luma component (Y) for brightness and two chrominance components (U and V) for color corresponds more closely to the human vision perception of colors than the standard RGB color model. Since the performance of a segmentation result is mostly evaluated by human eyes, we consider the YUV information more appropriate for image segmentation. The data for each image is then presented by a matrix of dimension 38400×5: There are 38400 pixels in each of the test images, and the information of each pixel point is described by the five aforementioned variables: x, y, Y, U and V.

The next question was: how to balance between the two types of information, position and color? When the information on position is given more weight in the process of segmentation, adjacent areas with distinct colors are more likely to be combined as one region. A contrary result would have happened when the information on color is given more weight. The weighting coefficient for the two types of information should therefore depend on the desired sizes of the resulting segmented regions. These desired sizes should reflect the true sizes of the shapes of the major objects in the image to be segmented. That is to say, the weighting coefficient should vary for each individual image. We introduced a scaling parameter α that serves as the weighting coefficient. This parameter is used to re-scale the x and y coordinates to x/α and y/α.

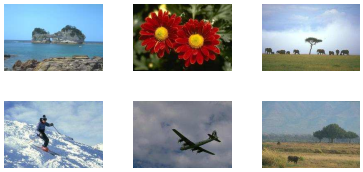

Figure 13: Test images from "The Berkeley Segmentation Dataset"

We applied SUP with dynamic temperature on the constructed data of five variables to cluster pixels. In the f function presented in (2), we replaced the Euclidean distance L^2 with the absolute distance L^1 to reduce the effect of possible large deviations in one dimension only. For each test image, various sets of parameter values for r and α were experimented, with r ranging from 60 to 120 and α from 8 to 20. Figure 14 presents the best segmentation results for each image, showing nice segmentations simply by the use of SUP. A combined approach of SUP with other segmentation techniques is therefore expected to be very promising for segmenting images.

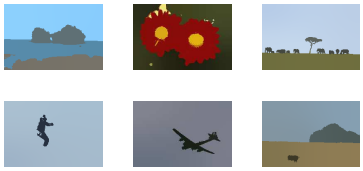

Figure 14: Image segmentation results of the test images in figure 13 by SUP

6 Discussion and Conclusion

The self-updating process is a simple, intuitive and powerful clustering algorithm. In the updating process, each element moves to a new position at each iteration. The new position depends on the influences the element receives from other elements. At the end of the process, every element reaches its equilibrium position without further movements. Elements that arrive the same position are considered to belong to the same cluster. This algorithm works straightforward. The convergence of the updating process is also proved.

Although SUP is not particularly developed to handle noise in the data, this ability comes naturally as a byproduct. Recall that elements in the data are clustered on the basis of mutual influence, which is defined to be larger when two elements are closer. As a result, noise data that are not close to other data points is bound to be isolated at the end of the updating process. The noise data points can therefore be easily identified. In Section 4.1 and Section 5.1, we showed the strength of SUP in separating noise. This strength offers a great advantage to the use of SUP, especially when the noise level in the data is substantially high.

Compared to k-means algorithm that minimizes the sum of the within-cluster variations, SUP is not one of such clustering algorithms that optimize certain criterion functions. Although it is often appealing to have clustering results that represent specific statistical terms, such as the solution of k-means algorithm nicely represents "the minimizer of the sum of the within-cluster variations", there are times when these terms are not what we truly seek in the data. Section 4.2 presented one example, in which the criterion used by k-means algorithm does not conform to the structure of the data. Under such a situation, the minimizer of the sum of the within-cluster variations can not reveal the true data structure. Poor clustering results by k-means algorithm presented in Section 4.2 verified this point: the use of criterion functions for clustering is sometimes inappropriate.

The self-updating process is in some sense a slow version of agglomerative hierarchical clustering: In SUP, two elements are merged gradually instead of at once. As one weakness of hierarchical clustering is that early mistakes cannot be corrected, slowing down the merging process especially at the beginning stage can very often reduce the chances of making mistakes. Later we realized that the connection between agglomerative hierarchical clustering and SUP is even

closer: When the function f in (1) is generalized to be non-homogeneous in t, agglomerative hierarchical clustering with centroid linkage can be written as a special case of SUP. We write the non-homogeneous function f_t as

$$f_t(x_i^{(t)}, x_j^{(t)}) = \begin{cases} 1, & d(x_i^{(t)}, x_j^{(t)}) \leq r^{(t)}, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where the influential range $r^{(t)}$ changes at each iteration,

$$r^{(t)} = \min_{k \neq l \text{ and } x_k^{(t)} \neq x_l^{(t)}} ||x_k^{(t)} - x_l^{(t)}||.$$

This function f_t takes a positive value only when $i = j$, or when $x_i^{(t)} \neq x_j^{(t)}$ and the distance between $x_i^{(t)}$ and $x_j^{(t)}$ is the smallest among all non-zero pairwise distances. Using this f_t, SUP at the first iteration only updates the pair that has the smallest pairwise distance. Both elements of the pair are updated to the averaged position of the pair according to (1). At later iterations SUP only updates the two groups that have the smallest between-group distance. Each element in the two groups is updated to the averaged position of all elements in the two groups. That is to say, SUP using the f function in (4) creates an identical merging process to agglomerative hierarchical clustering using centroid linkage.

In addition to the exponential decay function in (2) that is used throughout this paper, and the function in (4) that generates a process identical to agglomerative hierarchical clustering using centroid linkage, SUP can turn into other clustering processes by the use of various types of f functions. When there is prior information about the data structure, it is often desirable to include the prior information in the clustering process. This can be done by incorporating such information into the function f, then SUP can turn into a model-based algorithm. We are interested in the formations of f functions for data with certain structures, such as the spiral data and data with specific probabilistic distributions. By incorporating information that characterizes some known data structure, the performance of SUP can be further enhanced.

A Proof of Lemmas

A.1 Proof of Lemma 2

Since

$$C_1 = \lim_{t \to \infty} C_1^{(t)},$$

for each i, there exists a sequence of $v_{1,i}^{(t)}$'s (exchange vertex indices if necessary), such that

$$\lim_{t \to \infty} v_{1,i}^{(t)} = v_{1,i},$$

where $v_{1,i}^{(t)}$ is a vertex of $C_1^{(t)}$. Since for any t and i,

$$v_{1,i}^{(t)} = x_k^{(t)}$$

for at least one k, there exists j, such that

$$x_j^{(t)} = v_{1,i}^{(t)}$$

for infinite many t's. Therefore, there exists an infinite time sequence t_n's, such that

$$x_j^{(t_n)} = v_{1,i}^{(t_n)} \qquad \forall n,$$

which leads to

$$\lim_{n\to\infty} x_j^{(t_n)} = v_{1,i}.$$

If $x_j^{(t)} = v_{1,i}^{(t)}$ except for any finite t, then equation (3) is established. Otherwise, there exists $j' \neq j$ and another infinite time sequence s_n's, such that

$$x_{j'}^{(s_n)} = v_{1,i}^{(s_n)} \qquad \forall n.$$

Without loss of generality, assume that $v_{1,i}^{(t)} = x_j^{(t)}$ or $x_{j'}^{(t)}$ for all $t > \tilde{t}$. From equation (1), if $x_j^{(s)} = x_{j'}^{(s)}$ for some s, $x_j^{(t)} = x_{j'}^{(t)}$ for all $t > s$. Therefore, for any $s > 0$, there exists $t > s$, such that $v_{1,i}^{(t)} = x_j^{(t)}$ and $v_{1,i}^{(t+1)} = x_{j'}^{(t+1)}$. We claim that this case, however, can never happen: when t is large enough, it is impossible that a data point inside the convex hull later becomes a new vertex, since it is closer to other points than the current vertex is. In the following we prove this claim only for the one dimensional case. For higher dimensional cases, one can project all data points onto to a proper line, and then make the same argument.

Without loss of generality, assume $v_{1,i} = 0$, $x_j^{(t)} \leq 0$, and $x_k^{(t)} > 0$ for $k \neq j$ or j'. If $x_{j'}^{(t+1)}$ later becomes the new vertex, then

$$\frac{\sum_{k=1}^{N} f(x_{j'}^{(t)}, x_k^{(t)}) \cdot x_k^{(t)}}{\sum_{k=1}^{N} f(x_{j'}^{(t)}, x_k^{(t)})} < \frac{\sum_{k=1}^{N} f(x_j^{(t)}, x_k^{(t)}) \cdot x_k^{(t)}}{\sum_{k=1}^{N} f(x_j^{(t)}, x_k^{(t)})}. \tag{5}$$

We claim that the inequality above is false. Since

$$f(x_{j'}^{(t)}, x_{j'}^{(t)}) = f(x_j^{(t)}, x_j^{(t)}) = 1 > f(x_{j'}^{(t)}, x_j^{(t)}),$$

$$
\begin{aligned}
&\frac{\sum_{k=1}^{N} f(x_{j'}^{(t)}, x_k^{(t)}) \cdot x_k^{(t)}}{\sum_{k=1}^{N} f(x_{j'}^{(t)}, x_k^{(t)})} \\
= \; &\frac{x_{j'}^{(t)} + f(x_{j'}^{(t)}, x_j^{(t)}) \cdot x_j^{(t)} + \sum_{k \neq j, j'} f(x_{j'}^{(t)}, x_k^{(t)}) \cdot x_k^{(t)}}{\sum_{k=1}^{N} f(x_{j'}^{(t)}, x_k^{(t)})} \\
\geq \; &\frac{(1 + f(x_{j'}^{(t)}, x_j^{(t)})) \cdot \frac{x_{j'}^{(t)} + x_j^{(t)}}{2} + \sum_{k \neq j, j'} f(x_{j'}^{(t)}, x_k^{(t)}) \cdot x_k^{(t)}}{\sum_{k=1}^{N} f(x_{j'}^{(t)}, x_k^{(t)})},
\end{aligned}
$$

and

$$
\begin{aligned}
&\frac{\sum_{k=1}^{N} f(x_j^{(t)}, x_k^{(t)}) \cdot x_k^{(t)}}{\sum_{k=1}^{N} f(x_j^{(t)}, x_k^{(t)})} \\
= \; &\frac{x_j^{(t)} + f(x_j^{(t)}, x_{j'}^{(t)}) \cdot x_{j'}^{(t)} + \sum_{k \neq j, j'} f(x_j^{(t)}, x_k^{(t)}) \cdot x_k^{(t)}}{\sum_{k=1}^{N} f(x_j^{(t)}, x_k^{(t)})} \\
\leq \; &\frac{(1 + f(x_{j'}^{(t)}, x_j^{(t)})) \cdot \frac{x_{j'}^{(t)} + x_j^{(t)}}{2} + \sum_{k \neq j, j'} f(x_j^{(t)}, x_k^{(t)}) \cdot x_k^{(t)}}{\sum_{k=1}^{N} f(x_j^{(t)}, x_k^{(t)})}.
\end{aligned}
$$

To prove that equation (5) is false, it suffices to show that

$$\begin{aligned} & \frac{(1+f(x_{j'}^{(t)},x_j^{(t)}))\cdot\frac{x_{j'}^{(t)}+x_j^{(t)}}{2}+\sum_{k\neq j,j'}f(x_j^{(t)},x_k^{(t)})\cdot x_k^{(t)}}{\sum_{k=1}^{N}f(x_j^{(t)},x_k^{(t)})} \\ < & \frac{(1+f(x_{j'}^{(t)},x_j^{(t)}))\cdot\frac{x_{j'}^{(t)}+x_j^{(t)}}{2}+\sum_{k\neq j,j'}f(x_{j'}^{(t)},x_k^{(t)})\cdot x_k^{(t)}}{\sum_{k=1}^{N}f(x_{j'}^{(t)},x_k^{(t)})}. \end{aligned}$$

We claim the inequality above is true, and complete the proof of lemma 2 based on the followings:

(i) Since $x_j^{(t)}$ is the vertex, $\|x_j^{(t)}-x_k^{(t)}\|>\|x_{j'}^{(t)}-x_k^{(t)}\|$ for all k, and hence $f(x_j^{(t)},x_k^{(t)})<f(x_{j'}^{(t)},x_k^{(t)})$.

(ii) Since $x_{j'}^{(t+1)}$ is the new vertex, it is is non-positive. Then

$$\begin{aligned} \frac{x_{j'}^{(t)}+x_j^{(t)}}{2} & < \frac{x_{j'}^{(t)}+f(x_{j'}^{(t)},x_j^{(t)})\cdot x_j^{(t)}}{1+f(x_{j'}^{(t)},x_j^{(t)})} \\ & < \frac{\sum_{k=1}^{N}f(x_{j'}^{(t)},x_k^{(t)})\cdot x_k^{(t)}}{\sum_{k=1}^{N}f(x_{j'}^{(t)},x_k^{(t)})}\leq 0. \end{aligned}$$

(iii) Since $\|x_j^{(t)}-v_{1,i}\|<\epsilon$, $x_j^{(t)}>-\epsilon$, and hence $\frac{x_{j'}^{(t)}+x_j^{(t)}}{2}>-\epsilon$. ϵ can be chosen arbitrary small so that $x_k^{(t)}>|\frac{x_{j'}^{(t)}+x_j^{(t)}}{2}|$ for $k\neq j,j'$.

(iv) The following lemma.

Lemma 4. *Suppose $x_1<0$, $x_k>|x_1|$ for all $k=2,\cdots,n$, $a_1=b_1>0$, and $a_k>b_k>0$ for all $k=2,\cdots,n$. If*

$$\frac{\sum_{k=1}^{n}a_kx_k}{\sum_{k=1}^{n}a_k}<0,$$

then

$$\frac{\sum_{k=1}^{n} a_k x_k}{\sum_{k=1}^{n} a_k} > \frac{\sum_{k=1}^{n} b_k x_k}{\sum_{k=1}^{n} b_k}.$$

Proof. Without loss of generality, assume that $x_1 = -1$, otherwise, we can divide x_k's by $|x_1|$. Let

$$c = \frac{\sum_{k=1}^{n} a_k x_k}{\sum_{k=1}^{n} a_k}.$$

Since c is the weighted average of x_k's and $x_k \geq -1$, $-1 < c < 0$. Define

$$c_k = \frac{a_k(x_k - c)}{c+1} \quad \forall k.$$

Note that $c_k > 0$ for $k \geq 2$, since $x_k > 0$ for $k \geq 2$. Then

$$\begin{aligned}
\sum_{k=1}^{n} c_k &= \sum_{k=1}^{n} \frac{a_k(x_k - c)}{c+1} \\
&= \frac{1}{c+1}\left(\sum_{k=1}^{n} a_k x_k - c\sum_{k=1}^{n} a_k\right) \\
&= \frac{1}{c+1}\left(\sum_{k=1}^{n} a_k x_k - \sum_{k=1}^{n} a_k x_k\right) \\
&= 0 \\
\Longrightarrow \sum_{k=2}^{n} c_k &= -c_1 \\
&= a_1.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\sum_{k=1}^{n} a_k x_k}{\sum_{k=1}^{n} a_k} &= \frac{-a_1 + \sum_{k=2}^{n} a_k x_k}{a_1 + \sum_{k=2}^{n} a_k} \\
&= \frac{-\sum_{k=2}^{n} c_k + \sum_{k=2}^{n} a_k x_k}{\sum_{k=2}^{n} c_k + \sum_{k=2}^{n} a_k} \\
&= \frac{\sum_{k=2}^{n} -c_k + a_k x_k}{\sum_{k=2}^{n} c_k + a_k}.
\end{aligned}$$

For each $k \geq 2$, it is obvious that

$$\frac{-c_k + a_k x_k}{c_k + a_k} > \frac{-c_k + b_k x_k}{c_k + b_k},$$

as $a_k > b_k$ means that the left-hand side puts more weight on x_k. Furthermore,

$$\begin{aligned}
\frac{-c_k + a_k x_k}{c_k + a_k} &= \frac{\frac{a_k(x_k - c)}{c+1} + a_k x_k}{\frac{a_k(x_k - c)}{c+1} + a_k} \\
&= \frac{a_k(x_k - c) + (c+1) a_k x_k}{a_k(x_k - c) + (c+1) a_k} \\
&= \frac{c a_k (x_k - 1)}{a_k (x_k - 1)} \\
&= c.
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{\sum_{k=1}^{n} b_k x_k}{\sum_{k=1}^{n} b_k} &= \frac{\sum_{k=2}^{n} -c_k + b_k x_k}{\sum_{k=2}^{n} c_k + b_k} \\
&< c \\
&= \frac{\sum_{k=1}^{n} a_k x_k}{\sum_{k=1}^{n} a_k}.
\end{aligned}$$

□

A.2 Proof of Lemma 3

Without loss of generality, assume that $x_i^{(t)}$ is the only data point that converges to $v_{i,1}$.

$$\frac{\sum_{j=1}^{N} f(x_i^{(t)}, x_j^{(t)}) \cdot x_j^{(t)}}{\sum_{j=1}^{N} f(x_i^{(t)}, x_j^{(t)})} = x_i^{(t+1)}$$

$$\Rightarrow \quad \frac{\sum_{j=1}^{N} f(x_i^{(t)}, x_j^{(t)}) \cdot (x_j^{(t)} - x_i^{(t+1)})}{\sum_{j=1}^{N} f(x_i^{(t)}, x_j^{(t)})} = 0$$

$$\Rightarrow \quad \sum_{j \neq i}^{N} f(x_i^{(t)}, x_j^{(t)}) \cdot (x_j^{(t)} - x_i^{(t+1)}) = x_i^{(t+1)} - x_i^{(t)}. \tag{6}$$

Since $x_i^{(t)}$ converges to $v_{i,1}$, $x_i^{(t+1)}$ and $x_i^{(t)}$ become arbitrarily close to each other when t is large enough. That is, the right-hand side of (6) goes down to zero. On the other hand, since $x_j^{(t)}$ does not converge to $v_{i,1}$ for $j \neq i$, there is a gap between $x_j^{(t)}$ and $x_i^{(t+1)}$. To force the left-hand side of (6) to be zero, $f(x_i^{(t)}, x_j^{(t)})$ must goes down to zero as well. This sketches the proof for lemma 3. The precise details are given in the following.

Because $x_j^{(t)}$ does not converge to $v_{i,1}$ for $j \neq i$, there exists $\epsilon > 0$, for any $t_0 > 0$, there exists $t > t_0$ such that $\|x_j^{(t)} - v_{i,1}\| > \epsilon$. In fact, $x_j^{(t)}$ can not go arbitrarily close to $v_{i,1}$ when t is large enough, otherwise the updating process will move $x_j^{(t)}$ and $x_i^{(t)}$ closer and closer to each other. That is, there exists $\epsilon_0 > 0$ and t_1 such that $\|x_j^{(t)} - v_{i,1}\| > \epsilon_1$ for all $t > t_1$. On the other hand, because $x_i^{(t)} \to v_{i,1}$, for any $\epsilon_2 > 0$, there exists t_2, such that $\|x_i^{(t)} - x_i^{(t+1)}\| < \epsilon_2$ for $t > t_2$.

Since $v_{1,i}$ is a vertex of the convex set C_1, there exists $x \in C_1$, such that the inner product of $\overrightarrow{v_{1,i}x}$ and $\overrightarrow{v_{1,i}y}$ is positive for any $y \in C_1$. Let

$$v_x = \frac{\overrightarrow{v_{1,i}x}}{\|\overrightarrow{v_{1,i}x}\|}.$$

There exists $\alpha > 0$ and $t_3 > t_1$ such that

$$\langle x_j^{(t)} - v_{1,i}, v_x \rangle \geq \alpha \|x_j^{(t)} - v_{1,i}\| \qquad \forall t > t_3 \text{ and } \forall j \neq i,$$

where $\langle , \rangle$ denotes the inner product. Take the inner product of both sides of

(6) with v_x, we have

$$\begin{aligned}
& \left\langle \sum_{j\neq i}^{N} f(x_i^{(t)}, x_j^{(t)}) \cdot (x_j^{(t)} - x_i^{(t+1)}), v_x \right\rangle \\
= & \sum_{j\neq i}^{N} f(x_i^{(t)}, x_j^{(t)}) \cdot \left\langle x_j^{(t)} - x_i^{(t+1)}, v_x \right\rangle \\
= & \sum_{j\neq i}^{N} f(x_i^{(t)}, x_j^{(t)}) \cdot \left(\left\langle x_j^{(t)} - v_{1,i}, v_x \right\rangle + \left\langle v_{1,i} - x_i^{(t+1)}, v_x \right\rangle \right) \\
\geq & \sum_{j\neq i}^{N} f(x_i^{(t)}, x_j^{(t)}) \cdot \alpha \| x_j^{(t)} - v_{1,i} \| \\
> & \alpha\epsilon_1 \sum_{j\neq i}^{N} f(x_i^{(t)}, x_j^{(t)})
\end{aligned}$$

for $t > t_3$, and

$$\left\langle x_i^{(t+1)} - x_i^{(t)}, v_x \right\rangle \leq \| x_i^{(t+1)} - x_i^{(t)} \| < \epsilon_2$$

for $t > t_2$. Therefore, for $t > \max(t_3, t_2)$,

$$\alpha\epsilon_1 \sum_{j\neq i}^{N} f(x_i^{(t)}, x_j^{(t)}) < \epsilon_2.$$

Since ϵ_2 can be arbitrarily small, the inequality above implies

$$\sum_{j\neq i}^{N} f(x_i^{(t)}, x_j^{(t)}) \to 0.$$

Since $f \geq 0$, $f(x_i^{(t)}, x_j^{(t)}) \to 0$ for all $j \neq i$.

References

Banfield, J. D. and Raftery, A. E. (1993). Model-based gaussian and non-gaussian clustering. *Biometrics*, 49(3):803–821.

Chen, C. H. (2002). Generalized association plots: Information visualization via iteratively generated correlation matrices. *Statistica Sinica*, 12(1):7–29.

Cheng, Y. Z. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.

Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631.

Fukunaga, K. and Hostetler, L. D. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537.

Hartigan, J. A. (1975). *Clustering algorithms*. Wiley series in probability and mathematical statistics. Wiley, New York,.

Lloyd, S. P. (1982). Least-squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1 of *Statistics*, pages 281–297.

Mcquitty, L. L. (1968). Multiple clusters, types, and dimensions from iterative intercolumnar correlational analysis. *Multivariate Behavioral Research*, 3(4):465–477.

Milligan, G. W. and Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179.

Pan, W. (2006). Incorporating gene functions as priors in model-based clustering of microarray gene expression data. *Bioinformatics*, 22(7):795–801.

Selim, S. Z. and Alsultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, 24(10):1003–1008.

Shen, Y. J., Sun, W., and Li, K. C. (2010). Dynamically weighted clustering with noise set. *Bioinformatics*, 26(3):341–347.

Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 63:411–423.

Tseng, G. C. and Wong, W. H. (2005). Tight clustering: A resampling-based approach for identifying stable and tight patterns in data. *Biometrics*, 61(1):10–16.

Wu, H. M., Tien, Y. J., and Chen, C. H. (2010). Gap: A graphical environment for matrix visualization and cluster analysis. *Computational Statistics & Data Analysis*, 54(3):767–778.